ACKNOWLEDGEMENT

I take this opportunity to express my sincere gratitude to all those who helped me in various capacities in undertaking this project and devising the report.

I was privileged to express my sense of gratitude to our respected teachers Mr. Saiket Ghosh, Miss. Subhashri Talukdar, whose unparalleled knowledge, moral fiber and judgment along with their know-how, was an immense support in completing the project.

I am grateful to Ms. Saiket Ghosh, the Head of Department, for the brainwave and encouragement given.

I take this opportunity also to thank our friends and contemporaries for their cooperation and compliance.

Amit Sarkar

TABLE OF CONTENT

INTRODUCTION

The In this paper, I introduce ‘AI Chatbot: Sam,’ a Natural Language Processing (NLP) powered system designed specifically to assist students at Balurghat College. Sam goes beyond a simple information portal; it aspires to be a valuable companion throughout your college journey. Built with the needs of Balurghat College in mind, Sam empowers students to find the information they need efficiently, saving them time and frustration. By providing easy access to college-related information, Sam can contribute to a smoother and more positive college experience for all Balurghat College students.

1.1 Motive :

The inspiration for this project stemmed from a common frustration shared by many Balurghat College students – the struggle to navigate a cluttered college website. Witnessing the challenges students faced in finding the right information, I recognized the need for a more efficient and user-friendly solution. This realization sparked the idea for "AI Chatbot: Sam," a system powered by NLP that directly addresses student queries, eliminating the time-consuming search for answers on a disorganized website. By offering an accessible and interactive platform, Sam aims to empower students and ultimately create a smoother college experience. This application aims to empower users to:

- Provide Immediate and Easy Access to Information to the students.
- Save time by quickly and directly giving answers to student’s query
- Improve comprehension by focusing on the most relevant information.

1.2 Scope :

This project, titled "AI Chatbot: Sam," aims to develop a Natural Language Processing (NLP) powered chatbot specifically designed to assist students of Balurghat College. The scope of the project encompasses the following:

- ***Information Retrival:*** Sam will be trained on a comprehensive dataset of college-related information relevant to Balurghat College. This includes details on academics (courses, registration, deadlines), campus life (clubs, events, resources), admissions and financial aid, and any other pertinent college information.

- ***Natural Language Processing:*** Sam will utilize NLP techniques to understand student queries in natural language. This allows students to ask questions in their own words, eliminating the need for specific keywords.

- ***Direct Answers:*** Sam will strive to provide direct and accurate answers to student queries, drawing from its internal knowledge base. This eliminates the need for students to sift through irrelevant information on the college website.

1.3 Software Requirements :

The development of this web application utilizes the following software components:

- **Operating System :** Linux, Windows 7 or higher.
- **Programming Language :** Python
- **NLP Library :** NLTK (Natural Language Toolkit)
- **Front End tool :** HTML, CSS
- **Web Framework :** Flask

- **Software Justification :**

Operating System:

As now a day's windows 7 or higher are more common in market, we have designed this software to support all these OS.

Programming Language :

Python is a programming language that is interpreted, object-oriented, and considered to be high-level too. Python is one of the easiest yet most useful programming languages which is widely used in the software industry. Its demand is growing at a very rapid pace due to its vast use cases in Modern Technological fields like Data Science, Machine learning, and Automation

Tasks. For many years now, it has been ranked among the top Programming languages.

NLP Library :

Natural Language Toolkit (NLTK) is one of the largest Python libraries for performing various Natural Language Processing tasks. From rudimentary tasks such as text pre-processing to tasks like vectorized representation of text – NLTK's API has covered everything.

PyTorch :

This project will leverage PyTorch, a powerful deep learning framework, to develop the core functionality of the AI chatbot, Sam. PyTorch will be used to train a neural network model on the college information dataset. This model will then be able to analyze student queries, understand their intent, and retrieve the most relevant information from its knowledge base to provide accurate answers.

Front End tool :

Hyper Text Markup Language (HTML) is the most extensively used in frontend. We used it to lay the groundwork for our website. It allows us to manage the source codes, track the changes that we've made in our code or even roll back to the previous state in a much convenient way. Cascading Style Sheets (CSS) is the language that is used to present the html document. We used it to create the page's layout, color, fonts, and style in our webpage.

Web Framework :

Flask is a lightweight and flexible micro web framework for Python that is designed to make getting started with web application development quick and easy. It provides

developers with the tools and libraries needed to build a web application, including support for routing, templates, and working with databases.

1.4 Hardware Requirements:

The configuration given below is the Hardware handled for the system development.

Processor	Intel Pentium (dual core) / AMD FX
Primary Memory (RAM)	2 GB or Above
Secondary Memory (Hard disk)	128 GB or Above
Monitor	COLOR, Any Size
Display card	Any
Mouse & Keyboard	Any Company

Implementation

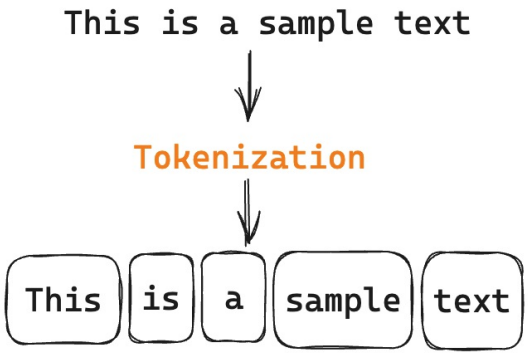
The implementation of "AI Chatbot: Sam" will involve several key steps. First, a comprehensive dataset of college information will be compiled, encompassing academics, campus life, admissions, and other relevant areas. Next, PyTorch will be used to train a deep learning model on this data. This model will learn to recognize patterns in student queries and retrieve corresponding information from the knowledge base. Finally, a user interface will be developed to facilitate natural language interaction between students and Sam. This could involve a text-based chat interface or potentially a voice-enabled system for an even more intuitive experience.

2.1 Natural Language Processing :

A core aspect of this project is Natural Language Processing (NLP), which allows Sam, the AI chatbot, to understand the intent behind student queries. NLP techniques like tokenization will be employed to break down student questions into individual words or phrases (tokens). Following tokenization, stemming will be used to reduce these tokens to their root forms. This process ensures Sam can comprehend questions regardless of variations in verb tenses or plurals, allowing for more accurate interpretation and retrieval of the most relevant information from its knowledge base.

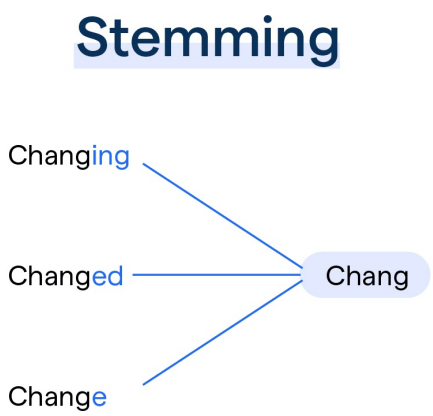
2.2 Tokenization :

In the context of "AI Chatbot: Sam," NLP Tokenization is the initial step where Sam breaks down student queries into understandable pieces. Imagine Sam dissecting a sentence like "What are the deadlines for course registration?" Tokenization would separate this into individual words ("What", "are", "the", "deadlines", "for", "course", "registration") or potentially even smaller units like punctuation marks. This allows Sam to analyze each element of the query and efficiently match it with the relevant information in its college knowledge base.



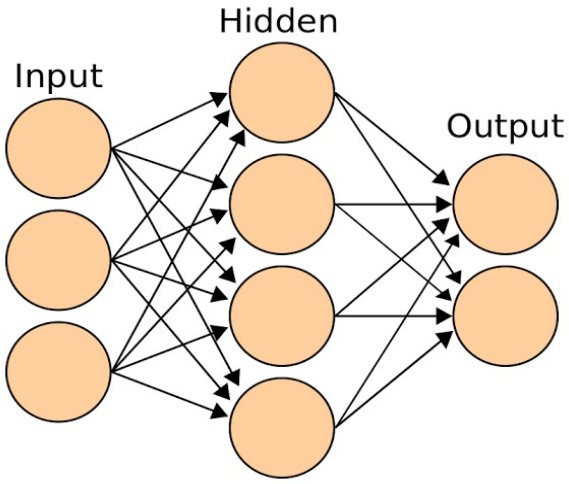
2.3 Stemming :

In the context of this project, NLP Stemming will play a role in processing student queries. Stemming refers to a technique that reduces words to their base or root form. For instance, "changing," "Changed," and "Chnage" would all be stemmed to "Chang." This helps Sam understand the core meaning of a student's question, even if they use variations of a word. While stemming can improve efficiency, it's important to note that it can sometimes lead to incorrect interpretations. We will carefully evaluate the trade-offs between accuracy and efficiency when implementing stemming in Sam's NLPprocessing.



2.4 SoftMax Algorithm :

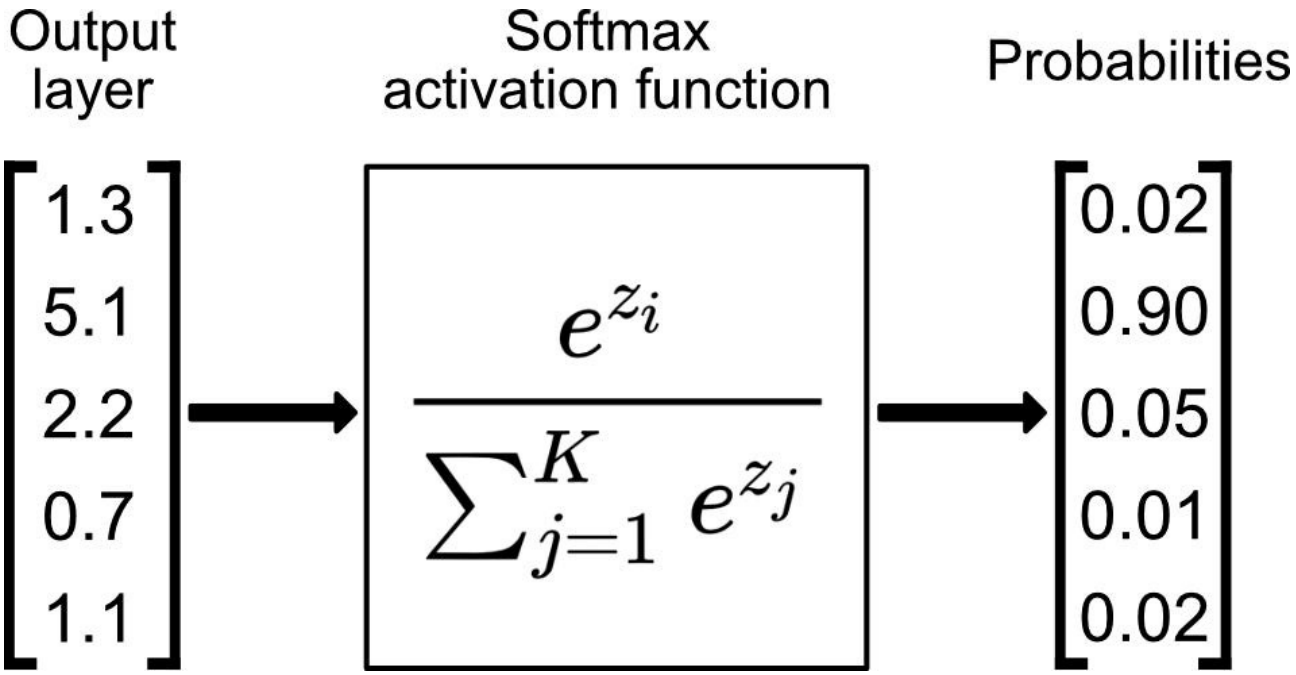
In this project, "AI Chatbot: Sam" utilizes Natural Language Processing (NLP) to understand the natural language students use in their queries. NLP techniques allow Sam to break down sentences, identify keywords, and grasp the intent behind a student's question. Combined with a Softmax Activation Function, Sam can translate this understanding into actionable responses. The Softmax function plays a crucial role by transforming the processed information into probabilities, enabling Sam to identify the most likely answer and deliver it with confidence.



The softmax function, often used in the final layer of a neural network model for classification tasks, converts raw output scores — also known as logits — into probabilities by taking the exponential of each output and normalizing these values by dividing by the sum of all the exponentials. This process ensures the output values are in the range (0,1) and sum up to 1, making them interpretable as probabilities.

The mathematical expression for the softmax function is as follows:Here, zi represents the input to the softmax function for class i, and the denominator is the sum of the exponentials of all the raw class scores in the output layer.Imagine a

neural network tasked with classifying images of handwritten digits (0-9). The final layer might output a vector with 10 numbers, each corresponding to a digit. However, these numbers don't directly represent probabilities. The softmax function steps in to convert this vector into a probability distribution for each digit (class).



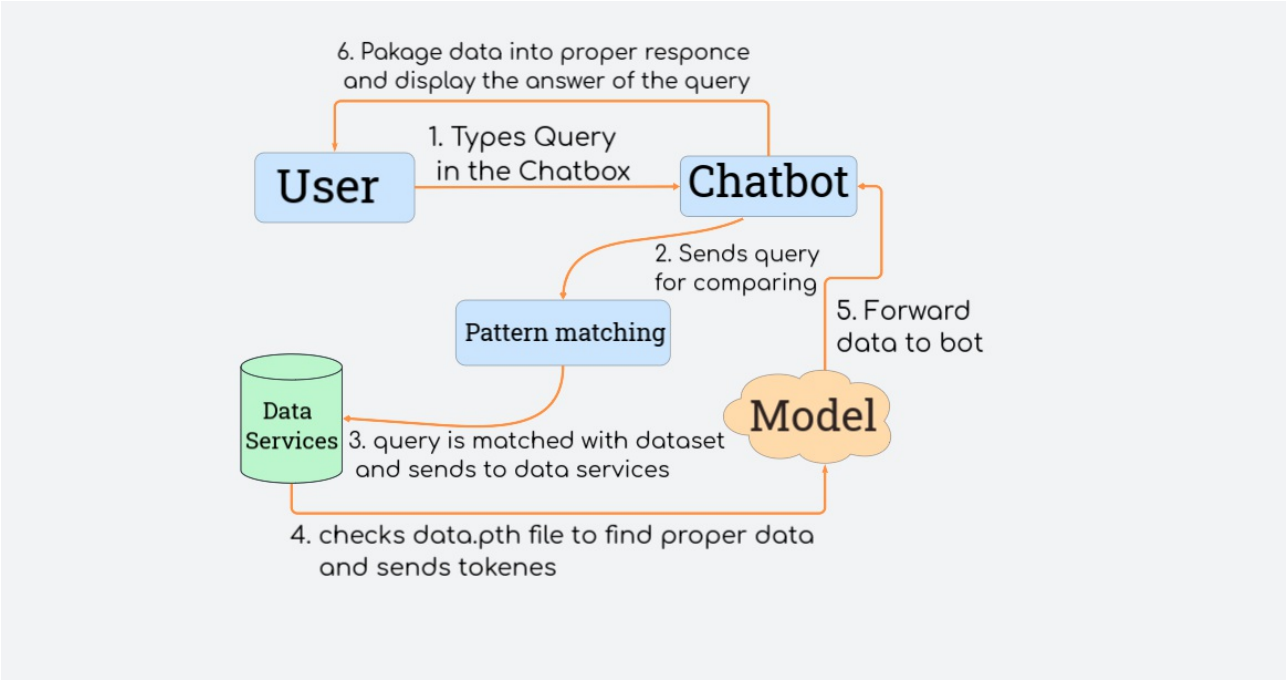
Input. The softmax function takes a vector z of real numbers, representing the outputs from the final layer of the neural network.

Exponentiation. Each element in z is exponentiated using the mathematical constant e (approximately 2.718). This ensures all values become positive.

Normalization. The exponentiated values are then divided by the sum of all exponentiated values. This normalization step guarantees the output values sum to 1, a crucial property of a probability distribution.

DESINING

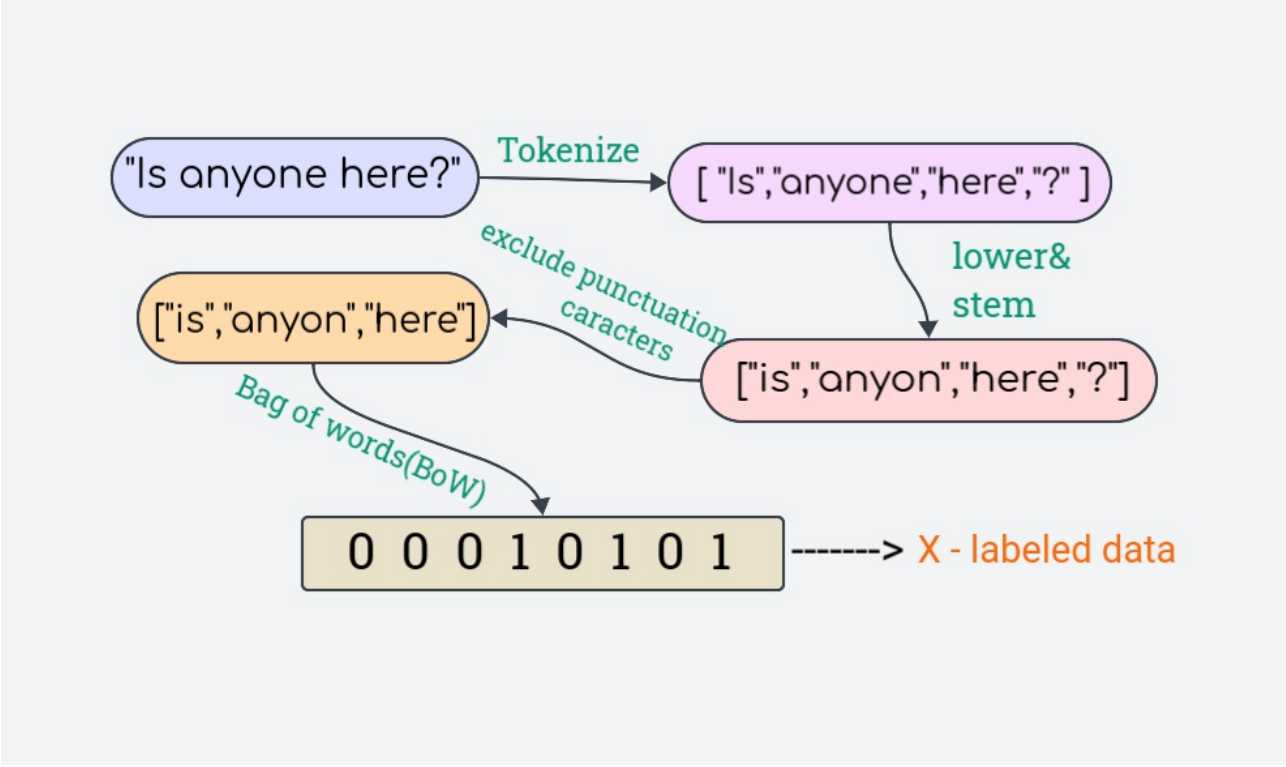
The design process for "AI Chatbot: Sam" prioritizes user experience and efficient information retrieval. This begins with collecting and organizing a comprehensive dataset encompassing all aspects of college life relevant to Balurghat College. Next, NLP techniques will be used to train Sam to identify the underlying intent behind student queries, regardless of how they're phrased. We'll then meticulously craft conversation flows to ensure Sam guides users towards the information they need in a clear and intuitive manner. Finally, Sam will be rigorously trained on the prepared dataset, followed by extensive testing to refine its ability to understand and respond to student queries accurately.



User types the query in the chatbox for responce, then the query is forwarded to chatbot**.Chatbot** further sends the query for pattern matchis step. Then the quey is matched with datasets and sends to the **Data Services**.The data file is checked where the tranning datas are stored, is it finds the match then it forwardes the propper data responce to the **Model**. After that the model generates the data to the bot and the chatbot pakage data into proper responce and display the message.

3.1 NLP Processing Pipeline :

A String is is tokenize for further process.Then all tokens are lowered and stemed ,we are excluding punctuation caracters.Then we use Bag of Words
Function to get a lebeled tranning data.



Tokenization: First, the data will be broken down into its fundamental units – words and phrases. This process, called tokenization, ensures the system recognizes individual elements for further processing.

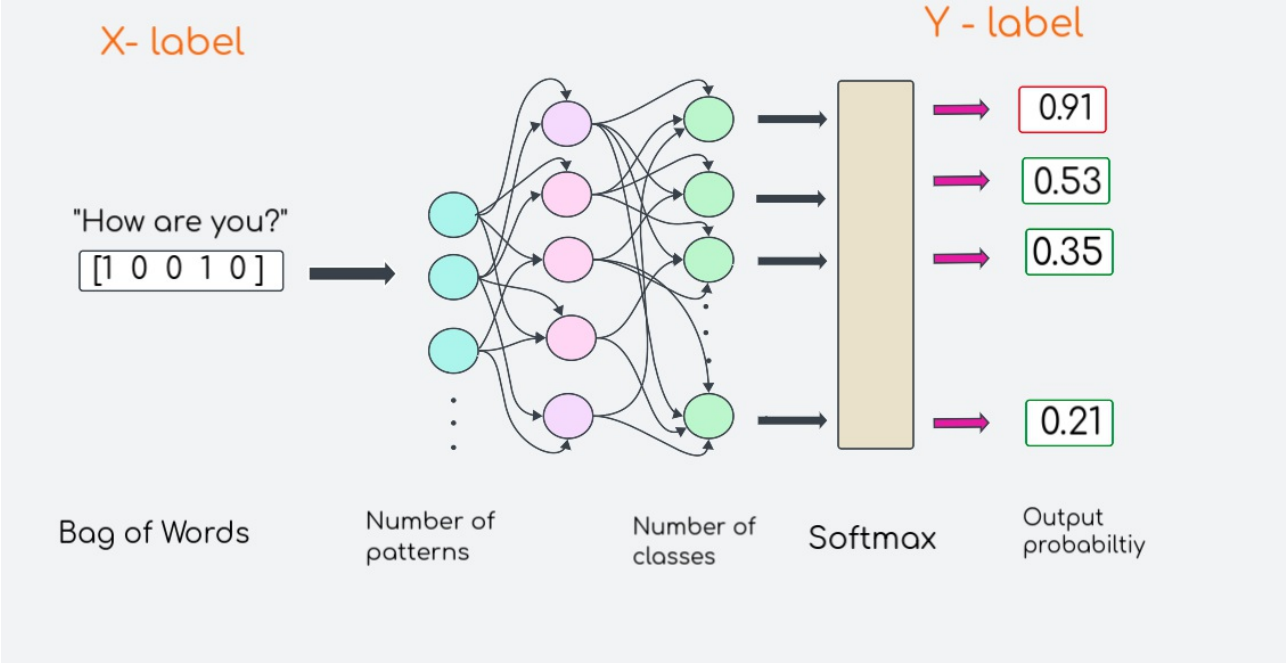
Stemming : Following tokenization, we may consider applying stemming. Stemming reduces words to their root forms, allowing Sam to capture the meaning even if encountering variations like "playing" and "played." This step can improve the model's ability to generalize and identify relevant information.

Punctuation Removal: Since punctuation characters don't hold inherent meaning in our context, they'll be excluded from the data. This simplifies the processing for Sam and prevents punctuation from influencing the model's understanding.

Bag-of-Words Representation: Finally, the processed data will be converted into a Bag-of-Words (BoW) representation. In a BoW, each unique word becomes a feature, and its frequency within the data determines its weight. This numerical representation allows the machine learning model to analyze the information and identify patterns for effective information retrieval.

3.2 Model Implementation:

At the heart of "AI Chatbot: Sam" lies a powerful NLP model implemented using PyTorch. This model will be trained on the meticulously curated dataset of college-related information specific to Balurghat College. The training process will involve feeding the model vast amounts of text data, including student queries, corresponding information from the dataset, and potentially pre-defined responses. Through this training, the model will learn to recognize patterns in language and map user queries to the most relevant information within its knowledge base. PyTorch's deep learning capabilities will allow the model to continuously improve its accuracy in understanding natural language and providing informative responses to student inquiries.



DEVELOPMENT PROCESS

This section details the development process for your text summarization web application using Flask, HTML, CSS, and NLTK for extractive summarization. It outlines the key stages involved and expands on each step for your project report.

4.1 Project Setup and Environment Configuration:

- Virtual Environment Creation:

To isolate project dependencies and avoid conflicts with system-wide Python installations, a virtual environment is created using tools like *venv* or *virtualenvwrapper*. This ensures a clean development environment with only the necessary libraries installed for this project.

- Dependency Installation:

The required libraries for the application are installed using *pip* within the activated virtual environment. These typically include:
Flask: A lightweight Python web framework for building web applications.
nltk: The Natural Language Toolkit library, providing functionalities for text processing tasks like tokenization and stop word removal.

PyTorch:train a neural network model on the college information dataset. This model will then be able to analyze student queries, understand their intent, and retrieve the most relevant information from its knowledge base to provide accurate answers.

- NLTK Resource Download:

Certain NLTK resources, like pre-trained tokenizers and stop word lists, are essential for the application's functionality. These resources are downloaded using *nltk.downloader* within your Python code. This ensures NLTK has the necessary data to perform its tasks effective

4.2 Flask Application Development:

- Flask App Initialization:

A Flask application instance is created using *app = Flask(__name__)*. This line initializes the core Flask application object that will handle routing, templating, and

other functionalities within your web application. The *__name__* variable holds the name of the current Python module, typically *app.py* in this case.

- Route Definition:

Routes are defined within the Flask application to map URLs to specific functions that handle user requests and generate responses. These routes are created using the *@app.route* decorator, specifying the URL path and the HTTP method (usually GET or POST) associated with the route.
A route for the main page (/) is typically defined to render the user interface template (*index.html*) where users can input text and get answers.

4.3 Function Implementation:

- Function Definition:

Function named “tokenize”,”stem”,”bag_of_words” from *snltk_utils.py* file is defined to handle the core text processing . they takes the user-submitted text and then tokenize, stem and give label acordingly.

- NLTK Integration:

The *text_summraizer* function leverages NLTK for various text processing tasks:

Sentence Tokenization:
NLTK's *sent_tokenize* function is used to break down the input text into individual sentences. This allows the application to process each sentence independently for summarization.

Punctuation Removal:
Common words like ","and "?" are removed from the sentences using NLTK's *stopwords* corpus. These stop words typically don't hold much meaning for labeling and can be safely removed to focus on content-rich words.

4.4 HTML Template Development :

- Template Structure:

HTML templates are created using HTML and logic control statements like if and for loops (if applicable). These templates define the user interface of the application. Two main templates are typically used:

- *index.html:* This template defines the layout for the main page, including a form for users to paste their text and specify the desired number of sentences for the summary. It also includes a submit button to trigger form submission.

4.5 CSS Integration :

- CSS File Creation:

A CSS file (*style.css*) can be optionally created to style the HTML elements within the templates. This allows you to define styles

CODE IMPLEMENTATION

This section provides a detailed breakdown of the core Python codes (*app.py* and *summarizer.py*) for the text summarization web application, elaborating on functionalities and logic for your project report.

5.1nltk_utils.py:

Imports:

import nltk

nltk.download('punkt')

» nltk:

This import brings in the Natural Language Toolkit library, essential for text processing tasks in our application.

- Nltk_utlis.py :
- **Imports:**

import numpy as np

import nltk

nltk.download('punkt')

from nltk.stem.porter import PorterStemmer

stemmer = PorterStemmer()

```
def tokenize(sentence):
    """
    split sentence into array of words/tokens
    a token can be a word or punctuation character, or number
    """
    return nltk.word_tokenize(sentence)


def stem(word):
    """
    stemming = find the root form of the word
    examples:
    words = ["organize", "organizes", "organizing"]
    words = [stem(w) for w in words]
    -> ["organ", "organ", "organ"]
    """
    return stemmer.stem(word.lower())


def bag_of_words(tokenized_sentence, words):
    """
    return bag of words array:
    1 for each known word that exists in the sentence, 0 otherwise
    example:
    sentence = ["hello", "how", "are", "you"]
```

```
    words = ["hi", "hello", "I", "you", "bye", "thank", "cool"]
    bog   = [  0 ,    1 ,    0 ,   1 ,    0 ,    0 ,      0]
    """
    # stem each word
    sentence_words = [stem(word) for word in tokenized_sentence]
    # initialize bag with 0 for each word
    bag = np.zeros(len(words), dtype=np.float32)
    for idx, w in enumerate(words):
        if w in sentence_words:
            bag[idx] = 1


    return bag
```

5.2 app.py

- **Imports:**

```
from flask import Flask, render_template, request, jsonify
from flask_cors import CORS
from chat import get_response
```

» Flask:

This import brings in the Flask framework, allowing us to create a web application with functionalities like routing and templating.

» render_template:

This function from Flask helps render HTML templates with dynamic data passed from the application

» request:

This object from Flask provides access to user input submitted through forms.

- Flask Application Initialization:

```
app = Flask(__name__)
```

» This line initializes a Flask application instance named *app*. The *__name__* variable holds the name of the current Python module. This is useful for various purposes within Flask, such as template name resolution.

- **Route Definition for Main Page (/):**

```
@app.route('/')
        def index():
                return render_template('index.html')
```

» The *@app.route('/')* decorator defines a route that handles requests to the root URL (/).

» The specified method is *GET*, indicating this route handles requests where the browser retrieves information from the server (e.g., loading a web page).

» The *index* function is associated with this route. It simply renders the *index.html* template using *render_template*. This template likely contains the user interface for text input and summary selection.

- **Route Definition for chatbot (/predict):**

```
@app.post("/predict")
def predict():
    text = request.get_json().get("message")
    # TODO: check if text is valid
    response = get_response(text)
    message = {"answer":response}
    return jsonify(message)
```

- This route definition is similar to the previous one, but with some key differences:

» The route is set to */result* which will likely be the target URL for the form submission in the *index.html* template.

» The *methods* argument is set to [*'POST'*], indicating this route handles form submission requests *(POST method*) where data is sent from the client (*user*) to the server (*application*).

TESTING

This section explores the testing for the text summarization web application.

6.1 Local testing:

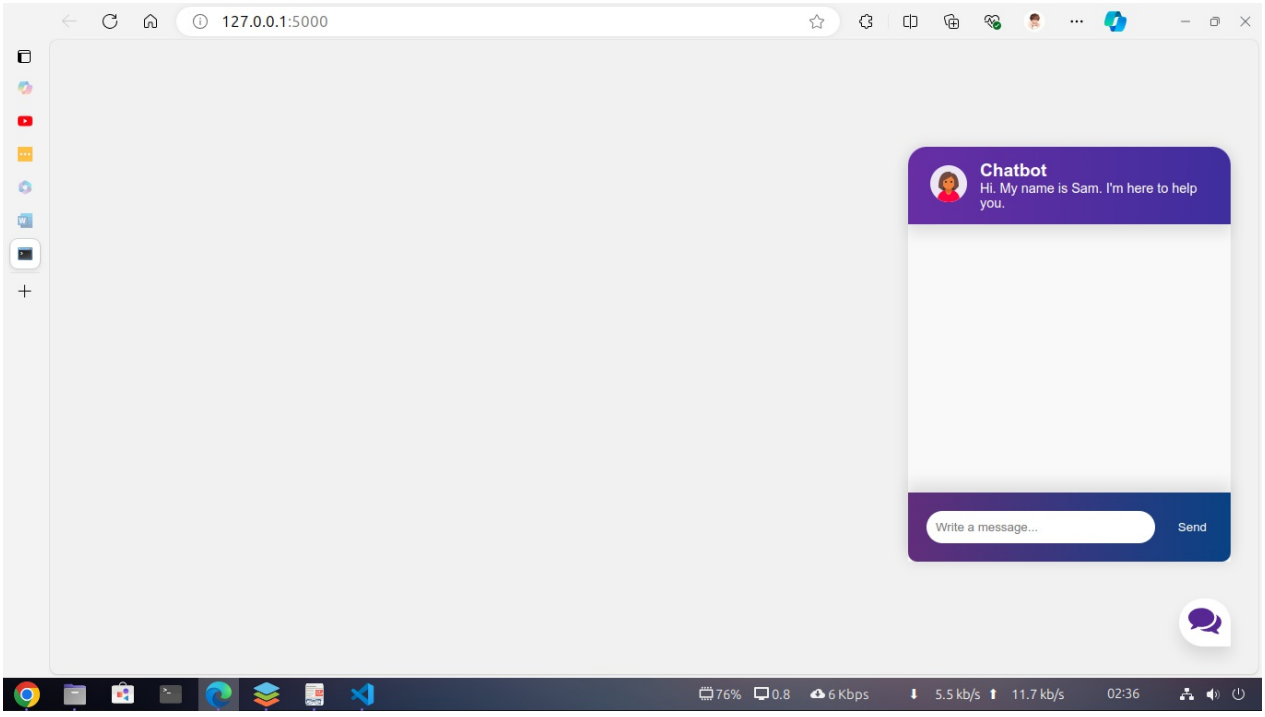
- It's crucial to thoroughly test the application locally during development. This involves running the Flask development server using flask run and accessing the application in a web browser (usually at http://127.0.0.1:5000/).
- Manual testing should be conducted to verify functionalities like:

» User interface elements function as expected.
» Form submission successfully triggers the summarization process.
» Generated summaries are relevant and concise, reflecting the main points of the input text.

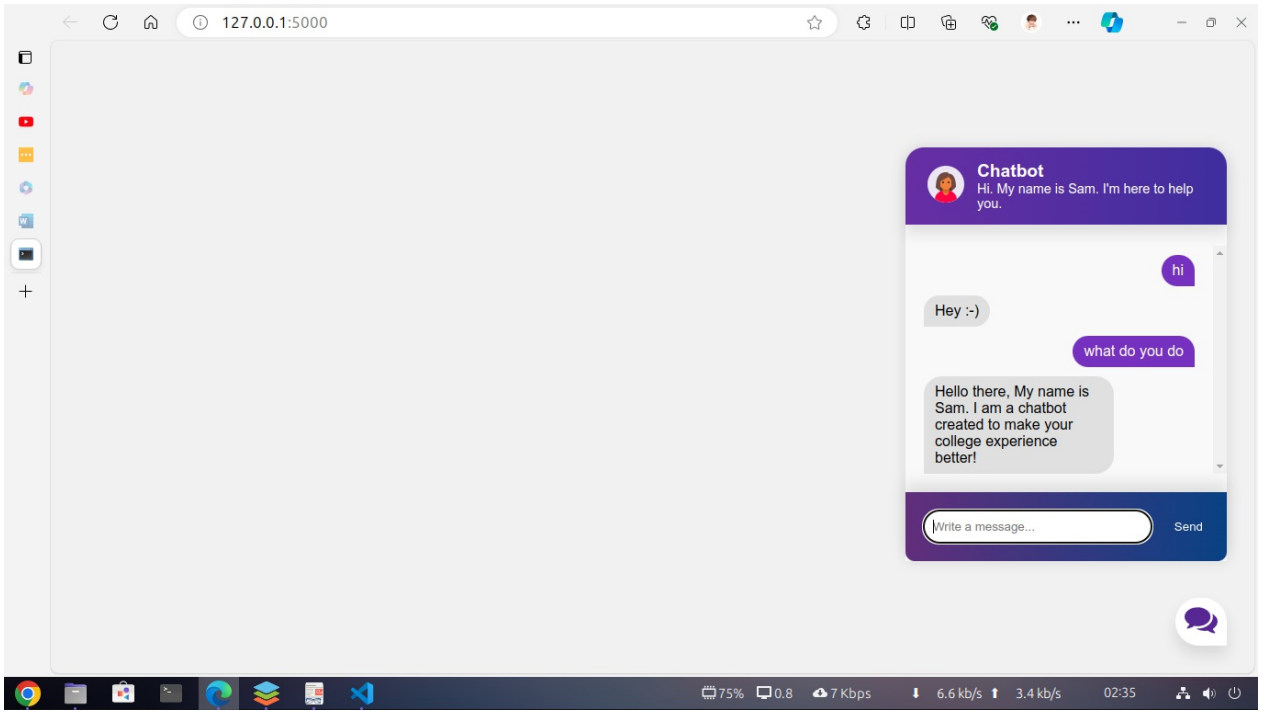
- Consider testing with various input scenarios: short texts, long texts, empty text, and texts with different complexities.

AI Chatbot : Sam

» Home page:



» messaging:



LIMITATIONS & FURTHER ENHANCEMENTS

7.1 Limitations

While "AI Chatbot: Sam" aims to be a valuable resource for Balurghat College students, there are some limitations to consider:

Limited Scope: Sam will focus solely on college-related information specific to Balurghat College. It won't be able to answer general questions or those requiring internet searches for broader topics.

Complexity: While Sam can answer direct questions, complex tasks requiring human interaction, like submitting applications or resolving intricate issues, may still require contacting college departments.

Understanding Nuance: Despite NLP advancements, Sam might struggle with heavily nuanced questions, sarcasm, or slang. Continued training and refinement will be necessary to improve its ability to handle these complexities of natural language.

Offline Dependence: Since Sam functions offline, its knowledge base may not reflect real-time updates or changes made on the college website. Regular updates will be crucial to ensure accuracy.

7.2 Further Enhancements

While "AI Chatbot: Sam" offers a promising solution for Balurghat College students, there's always room for improvement. Here are some exciting possibilities for future enhancements:

Integration with College Systems: Consider exploring the possibility of integrating Sam with existing college systems. This could allow students to perform basic tasks like registering for courses, checking grades, or scheduling appointments directly through Sam, further streamlining the college experience.

Multi-Lingual Support: To cater to a diverse student body, expanding Sam's capabilities to understand and respond in multiple languages would be a valuable

addition. This would make Sam more inclusive and accessible for international students or students who prefer to interact in their native language.

Sentiment Analysis and Personalized Recommendations: By incorporating sentiment analysis, Sam could gauge student satisfaction with responses and continuously improve its communication style. Additionally, the system could leverage user data to personalize recommendations for events, resources, or academic opportunities relevant to each student's interests and needs. Voice Assistant Integration: Future iterations could explore integrating Sam with a voice assistant platform, allowing students to interact with the chatbot through voice commands. This hands-free approach would enhance accessibility and convenience for students on the go.

Continuous Learning: Developing a mechanism for Sam to learn and adapt over time would be highly beneficial. This could involve allowing students to provide feedback on responses or enabling the chatbot to learn from new information added to its knowledge base. This continuous learning loop would ensure Sam remains up-to-date and provides the most accurate and relevant information.

CONCLUSION

"AI Chatbot: Sam" culminates in a valuable tool designed to empower Balurghat College students. Born from the frustration of navigating a cluttered website, Sam leverages NLP and the power of PyTorch to revolutionize information access. A robust NLP model, trained on a comprehensive dataset of college-specific information, forms Sam's core. This allows students to ask questions in natural language and receive direct, accurate answers, eliminating wasted time and frustration. By empowering students and fostering a positive college experience, Sam represents a significant step forward. Future developments like system integration and mobile accessibility hold promise for even greater impact. Ultimately, "AI Chatbot: Sam" embodies the potential of NLP and deep learning to transform information access within educational institutions, paving the way for a smoother and more empowered college journey for Balurghat College students.

REFERENCES

- www.tutorialspoint.com/
- www.javatpoint.com
- www.geeksforgeeks.org
- www.youtube.com
- https://flask.palletsprojects.com/
- www.nltk.org
- https://chatbotsmagazine.com/